Getting Started with zzlongplot

Introduction

`zzlongplot` is an R package designed for creating flexible and customizable visualizations of longitudinal data. Whether you are working with clinical trial data, repeated measures, or any data with a time- or visit-dependent structure, `zzlongplot` simplifies the process of creating grouped, faceted, and error-represented plots.

This vignette provides an overview of the package and demonstrates its key functionality through examples.

Installation

To install the `zzlongplot` package from GitHub, use the following commands:

```
install.packages("devtools") # If not already installed
devtools::install_github("your-username/zzlongplot")
```

Make sure to load the package before use:

```
library(zzlongplot)
```

Example Data

We will use synthetic datasets to demonstrate the functionality of `zzlongplot`.

Continuous x-axis

```
# Generate example data
df_cont <- data.frame(
  rid = rep(1:10, each = 3),
  visit = rep(c(0, 1, 2), times = 10),
  measure = rnorm(30, mean = 50, sd = 10),
  group = rep(c("A", "B"), length.out = 30)
)
```

Categorical x-axis

```
# Generate example data
df_cat <- data.frame(
  rid = rep(1:10, each = 3),
  visit = rep(c("baseline", "month1", "month2"), times = 10),
  measure = rnorm(30, mean = 50, sd = 10),
  group = rep(c("A", "B"), length.out = 30)
)
```

Using `lplot`

The primary function in `zzlongplot` is `lplot`, which generates the desired plots. This function combines observed values and change values into a single framework.

Observed Plot with Continuous x-axis

```
# Create observed plot
plot_obs <- lplot(
  df_cont,
  form = measure ~ visit | group,
  zeroval = 0,
  xlab = "Visit",
  ylab = "Measure",
  title = "Observed Values Over Time"
)
print(plot_obs)
```

Observed Plot with Categorical x-axis

```
# Create observed plot with categorical x-axis
plot_cat <- lplot(
  df_cat,
  form = measure ~ visit | group,
  zeroval = "baseline",
  xlab = "Visit",
  ylab = "Measure",
  title = "Observed Values Over Time (Categorical)"
)
print(plot_cat)
```

Combined Observed and Change Plots

To display both observed and change plots side-by-side:

```
# Combine observed and change plots
plot_both <- lplot(
  df_cont,
  form = measure ~ visit | group,
  zeroval = 0,
  ytype = "both",
  title = "Observed and Change Values"
)
print(plot_both)
```

Customizing Plots

Error Representation

Choose how to represent uncertainty using the `etype` argument: - `"bar"`: Error bars - `"band"`: Error ribbons

```r
# Observed plot with error bars
plot_error_bar <- lplot(
  df_cont,
  form = measure ~ visit | group,
  zeroval = 0,
  etype = "bar",
  title = "Observed Values with Error Bars"
)
print(plot_error_bar)

# Observed plot with error ribbons
plot_error_band <- lplot(
  df_cont,
  form = measure ~ visit | group,
  zeroval = 0,
  etype = "band",
  title = "Observed Values with Error Bands"
)
print(plot_error_band)
```

Faceting

Faceting allows stratified visualizations. Use the `facet_form` argument to specify row and column facets.

```r
# Add a new variable for faceting
df_cont$site <- rep(c("Site 1", "Site 2"), length.out = nrow(df_cont))

# Create faceted plot
plot_facet <- lplot(
  df_cont,
  form = measure ~ visit | group,
  facet_form = ~ site,
  zeroval = 0,
  title = "Faceted Observed Values"
)
print(plot_facet)
```

Helper Functions

`zzlongplot` includes several helper functions that work behind the scenes but can also be used independently.

`compute_stats`

This function computes summary statistics for observed and change values.

```r
# Compute statistics
stats <- compute_stats(df_cont, "visit", "measure", "group", "rid", 0)
print(stats)
```

`generate_plot`

This function creates a ggplot object based on the computed statistics.

```r
# Generate a plot
plot <- generate_plot(
  stats,
  x = "visit",
  y = "mn",
  grp = "grp",
  etype = "band",
  xlab = "Visit",
  ylab = "Measure",
  title = "Generated Plot"
)
print(plot)
```

Summary

`zzlongplot` simplifies the visualization of longitudinal data, providing robust tools for observed and change plots, error representations, grouping, and faceting. Its intuitive formula-based interface ensures ease of use, while customization options cater to advanced use cases.

We encourage users to explore the package and contribute via the GitHub repository.

For further details, refer to the reference manual.